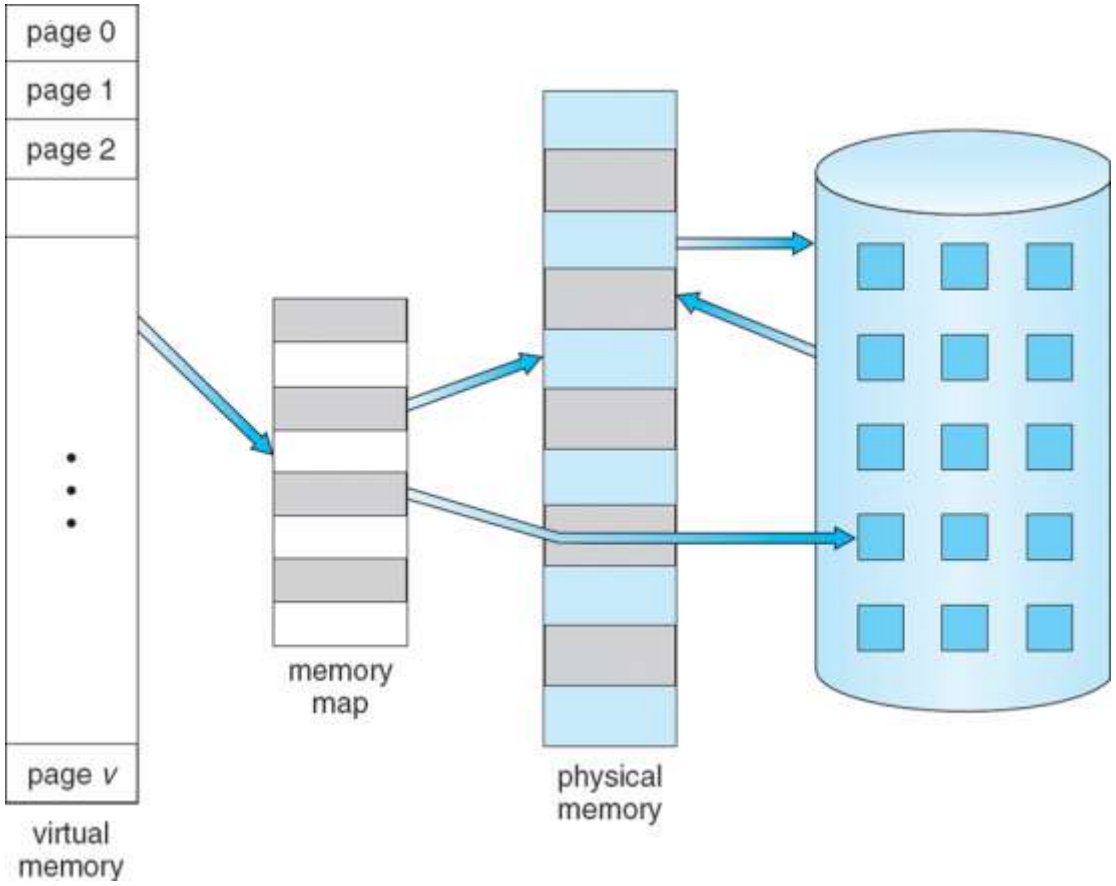
Virtual Memory

Sistem Operasi

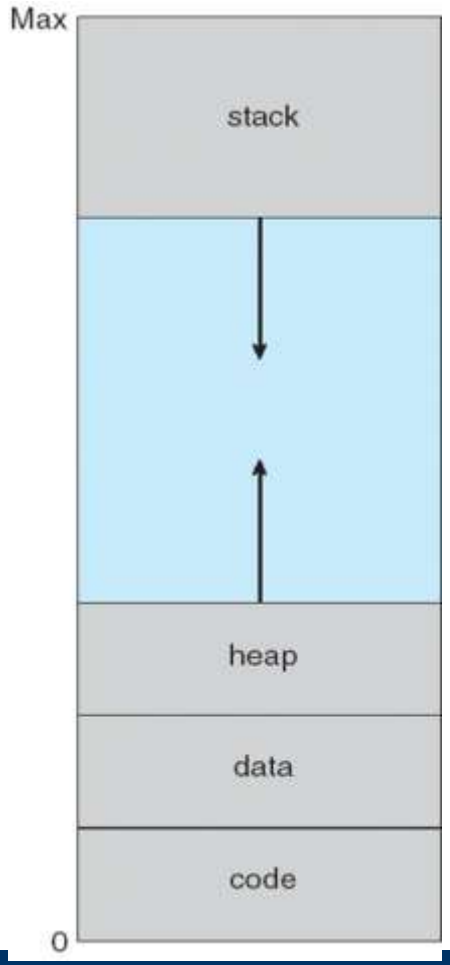
Virtual Memory

- **Tidak bisa** semua memory logik dipetakan semuanya ke memory fisik, walau **dynamic loading** bs melakukannya
- Memori virtual merupakan suatu teknik yang memisahkan antara memori logis dan memori fisiknya.
- Hanya bagian dari program yg **perlu** saja, berada di memory untuk eksekusi
- Logical address space bisa **lebih besar** daripada physical address space
- Memperbolehkan virtual address spaces pada VM untuk **disharing** oleh beberapa processes
- Bisa jadi hanya **beberapa bagian** dari memori logik yang berada di memori fisik, sisanya di harddisk.

Virtual Memory That is Larger Than Physical Memory



Virtual-address Space



Virtual Address Space yang memiliki Hole disebut **Sparse Address Space**

Program yg tidak perlu di Memory Utama

- Program-program (kode2) yg digunakan sbg error handling, yg jarang digunakan karena jarang terjadi
- Array, list, atau tabel yg kapasitasnya tidak terpakai semuanya
- Fungsi-fungsi yg tidak dipakai
- Program-program yang tidak digunakan scr real time

Virtual Memory

- Konsep memori virtual yang dikemukakan Fotheringham pada tahun 1961 pada sistem komputer Atlas di Universitas Manchester, Inggris:

  *“ Kecepatan maksimum eksekusi proses di memori virtual dapat sama, tetapi **tidak pernah melampaui** kecepatan eksekusi proses yang sama di sistem tanpa menggunakan memori virtual."*

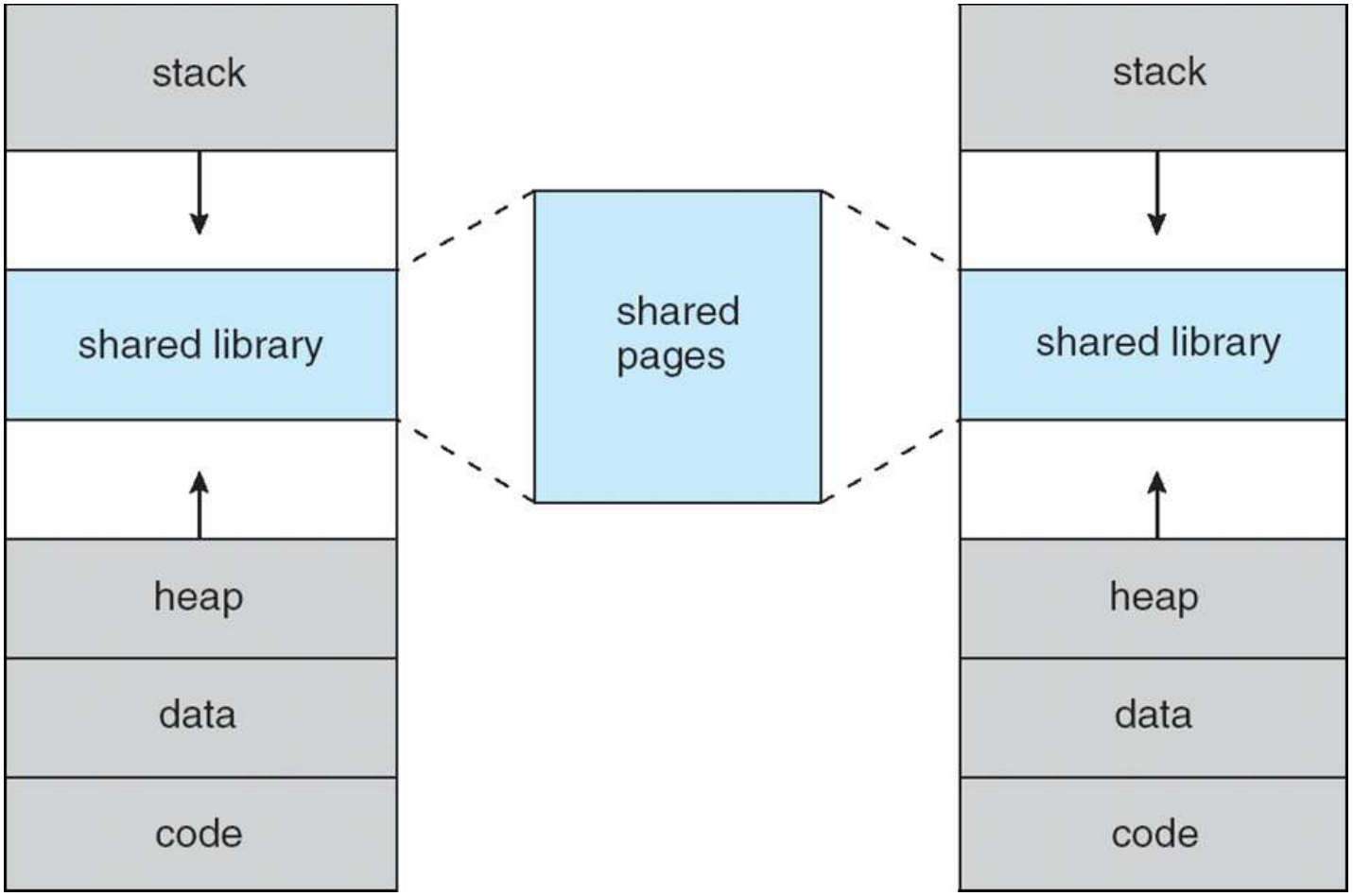
Keuntungan Virtual Memory

- Lalu lintas I/O menjadi rendah.
- Berkurangnya total memori fisik yang dibutuhkan.
- Meningkatnya respon, karena tidak deadlock.
- Bertambahnya jumlah user yang dapat dilayani.
- Memori virtual melebihi daya tampung dari memori utama yang tersedia.

Implementasi Virtual Memory

- Virtual Memory digunakan pada:
  - multiprograming
- Memori virtual dapat dilakukan dengan cara:
  - Demand paging

Shared Library Using Virtual Memory

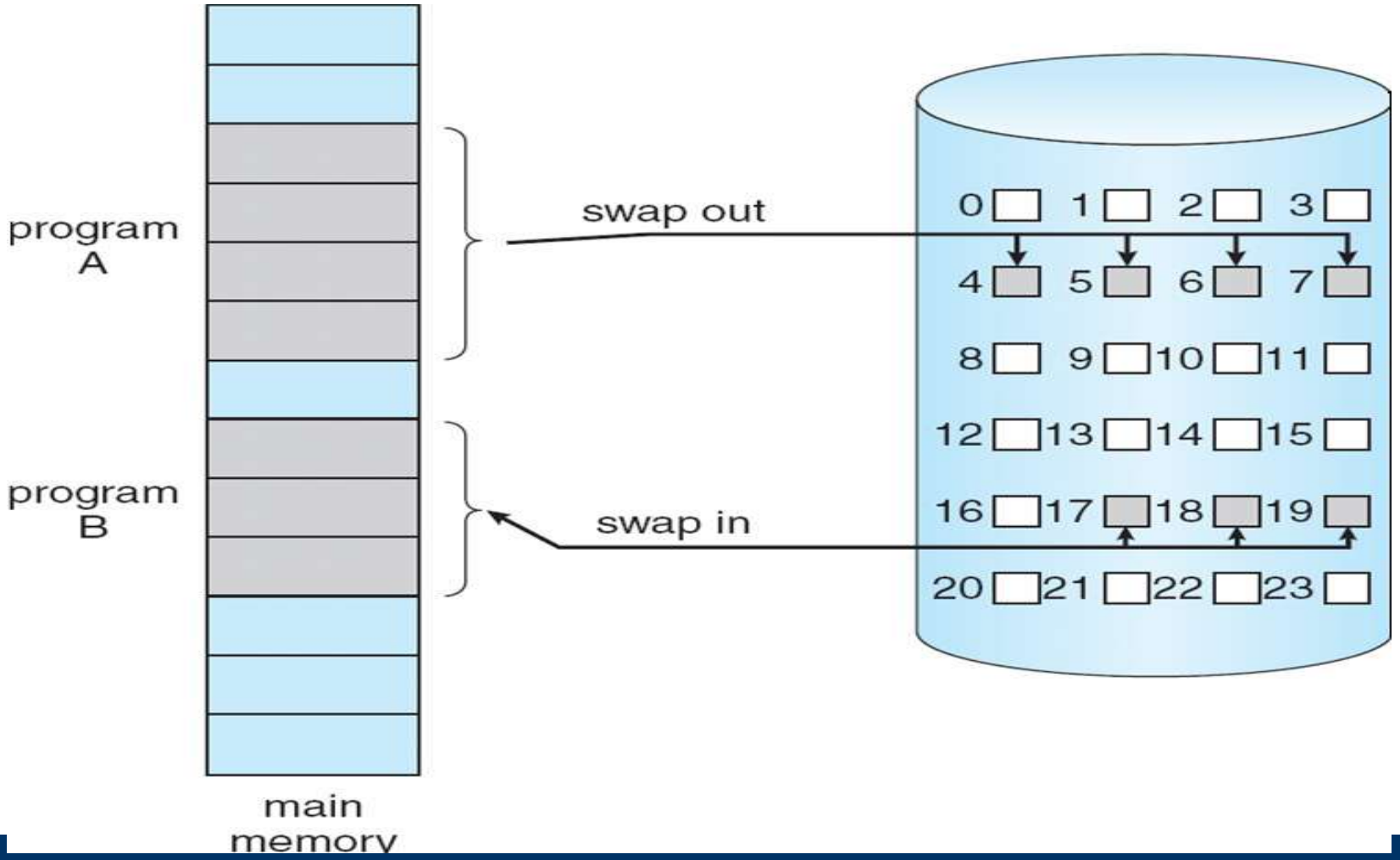


Sharing antar proses bisa diciptakan dgn **fork()**

Demand Paging

- Tidak semua program harus diload semua ke memory fisik
  - Hanya yg diperlukan saja
- Permintaan pemberian page menggunakan **swapping**.
- Page hanya akan di-swap ke memori utama jika benar-benar diperlukan.
- Program swapper yg digunakan:
  - **Lazy swapper** – tidak pernah swap page kedalam memory sampai page benar-benar diperlukan
  - Istilah Swapper berarti memanipulasi seluruh proses, sehingga swapper yang khusus berhubungan dengan pages bernama **pager**

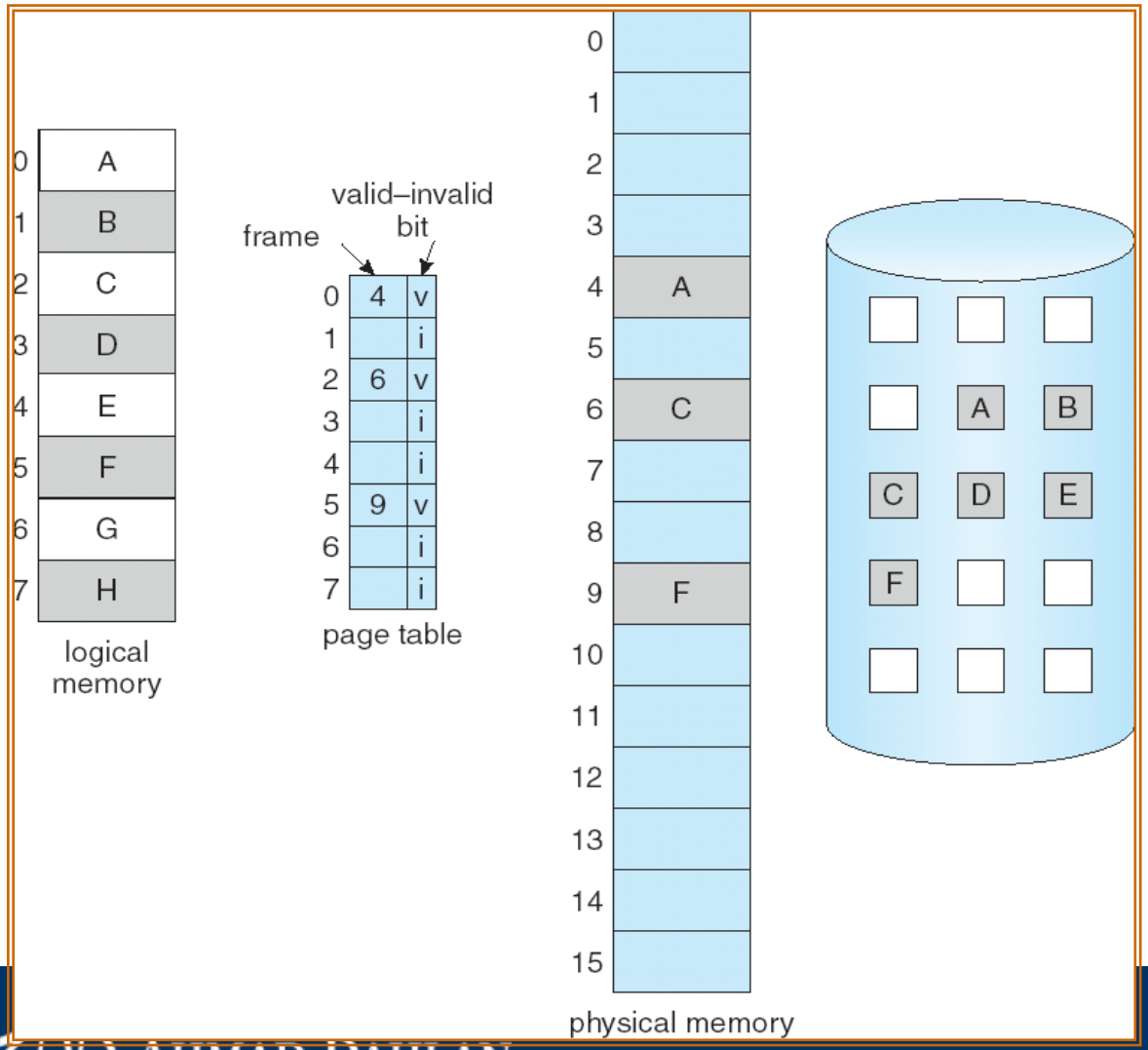
Transfer of a Paged Memory to Contiguous Disk Space



Demand Paging

- Pager hanya akan men-swap in dan out page yg dibutuhkan saja, tidak semuanya!
- Jadi, jika page dibutuhkan $\Rightarrow$ reference to it, tapi blm tentu semua diload ke memory fisik
- Butuh dukungan perangkat keras, yaitu:
  - Page-table: "valid-invalid bit"
    - Valid ("1") -> pages berada di memori fisik semuanya, atau pages ada, tapi tidak semuanya, sebagian masih berada di disk.
    - Invalid ("0") -> pages tidak ada di memory fisik.
  - Memori sekunder, untuk menyimpan proses yang belum berada di dalam memori fisik.
- Jika proses mengakses lokasi page yg valid, maka proses akan berjalan normal.
- Jika mengakses yg invalid, maka perangkat keras akan menjebaknya ke Sistem Operasi (**page fault**).

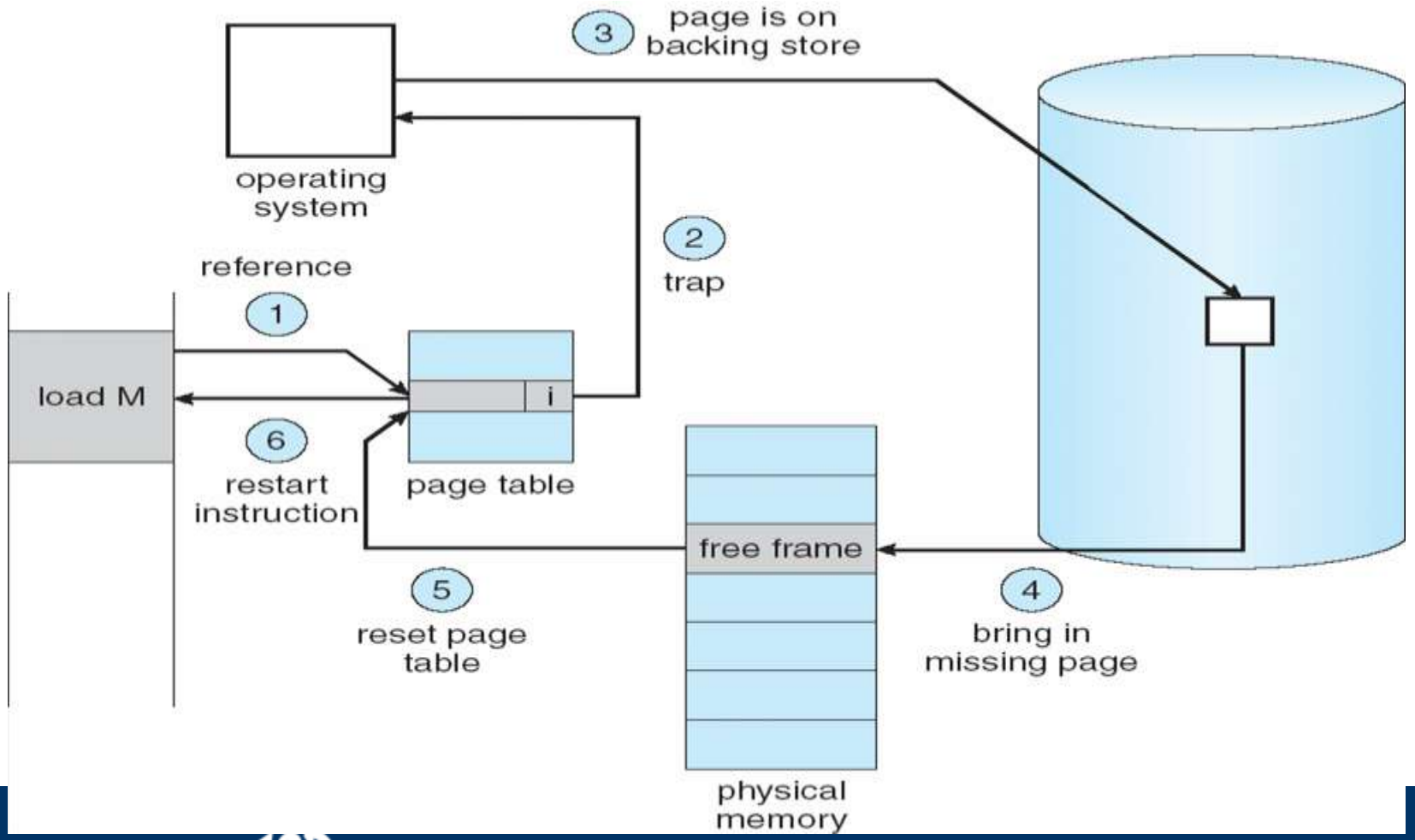
Page Table When Some Pages Are Not in Main Memory, but in HDD



Page Fault

- Jika ada referensi ke sebuah page, ternyata pagenya tidak ada (invalid), maka akan ditrap oleh OS, dan menghasilkan: **page fault**
- Untuk menangani page fault menggunakan prosedur berikut:
  - Memeriksa tabel internal (biasanya ada dlm PCB) unt menentukan valid atau invalid
  - Jika invalid, proses di suspend, jika valid tapi proses belum dibawa ke page, maka kita bawa page ke memory.
  - Cari sebuah frame bebas (free frame).
  - Jadwalkan operasi sebuah disk untuk membaca page tersebut ke frame yang baru dialokasikan.
  - Saat pembacaan selesai, ubah validation bit menjadi “1” yang berarti page telah ada di memory.
  - Ulangi lagi instruksi program yg ditrap td dari awal sehingga bisa berjalan dgn baik.

Steps in Handling a Page Fault



Yang dibutuhkan oleh Demand Paging

- Page Tabel yg berisi valid dan invalid bit
- Secondary Memory yg digunakan untuk menyimpan memory yg belum tertampung di main memory
  - Dikenal dng nama: swap file (linux) atau pagefile.sys (win)

Yg terjadi saat Page Fault

- Ditangkap oleh Sistem Operasi.
- SO menyimpan register user dan proses.
- Tetapkan bahwa interupsi merupakan page-fault.
- Periksa bahwa referensi page adalah valid dan kemudian tentukan lokasi page pada disk.
- Baca disk, cari frame kosong.
- Selama menunggu pencarian, alokasikan CPU ke proses lain dengan menggunakan penjadwalan CPU.
- Jika pencarian selesai, terjadi interupsi dari disk bahwa I/O selesai.

Yg terjadi saat Page Fault (2)

- SO menyimpan juga register dan status proses untuk pengguna/proses yang lain.
- Tentukan bahwa interupsi skrng berasal dari disk.
- Lakukan pengubahan page table bahwa page telah berada di memory.
- Tunggu CPU selesai dari proses yang tadi.
- Kembalikan register user, status proses, page table, dan resume instruksi proses yg td interupsi.

Page fault

- Tidak semua langkah diperlukan pada tiap kasus, ada 3 komponen utama yg pasti terjadi:
  - Melayani interrupt page fault
  - Baca dan load page dari disk ke memory
  - Restart proses
- Pada sistem demand paging, sebisa mungkin kita jaga agar tingkat page-fault nya rendah.

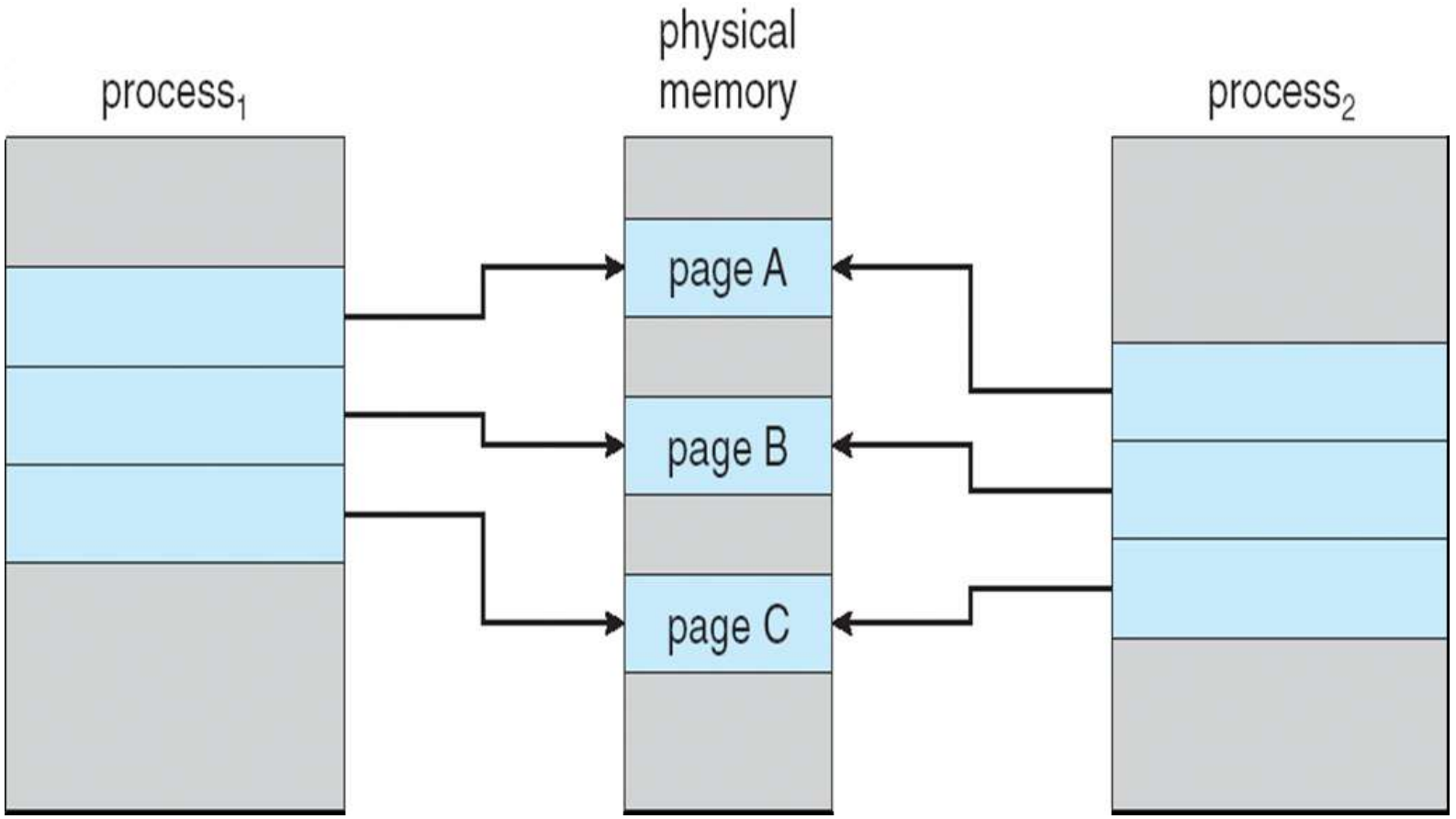
VM untuk Process Creation

- Karena diperlukan untuk menggandakan proses (process creation), maka harus diketahui mana page kosong yang akan dialokasikan.
  - Menggunakan fork()
- Sistem operasi biasanya menggunakan teknik “zero-fill-on-demand” untuk mengalokasikan page tersebut pada awalnya.

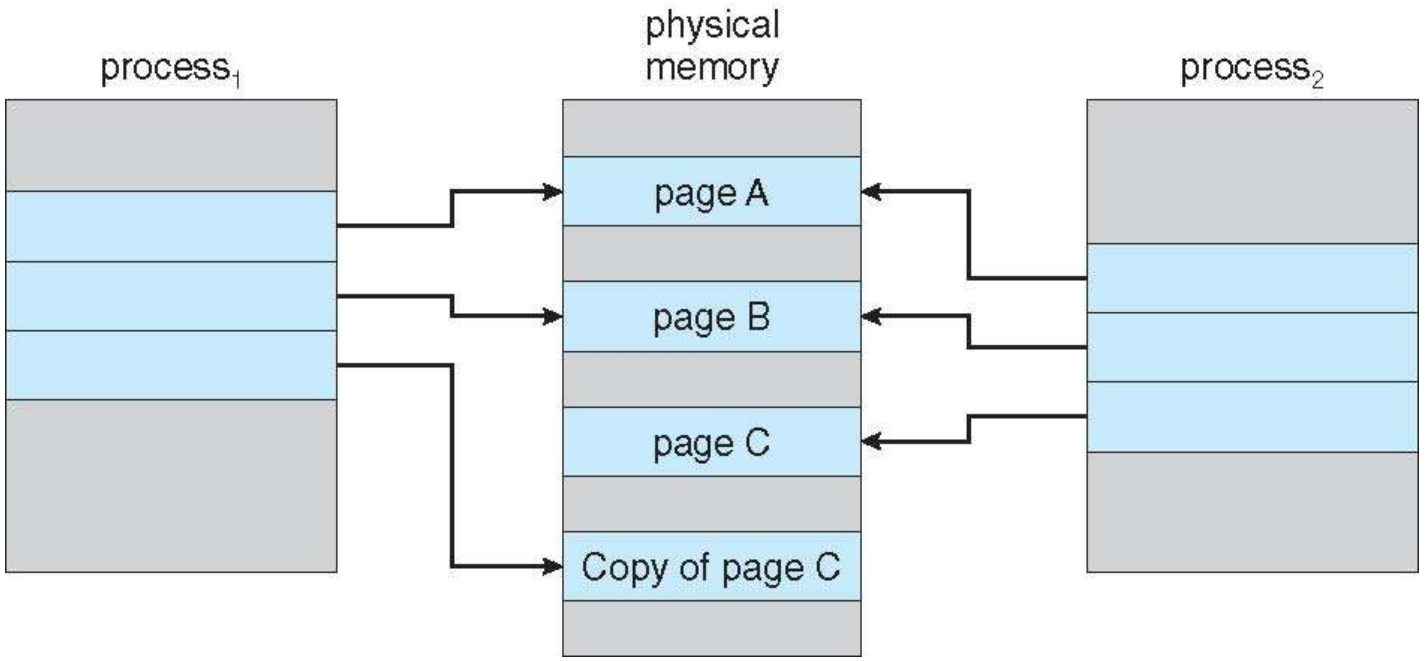
Cara Process Creation: Copy-on-write

- Pada copy-on-write, mengizinkan proses parent dan child menginisialisasikan **page yang sama** pada memori.
- Jika proses menulis pada sebuah page yang disharing, maka dibuat juga salinan dari page tersebut.
- Dengan menggunakan teknik **copy-on-write**, terlihat jelas bahwa hanya page yang diubah oleh proses child dan parent disalin. Sedangkan semua page yang tidak diubah bisa dibagikan ke proses child dan parent.
- Teknik copy-on-write sering digunakan oleh beberapa sistem operasi saat menggandakan proses. Diantaranya adalah Windows 2000, Linux, dan Solaris 2.

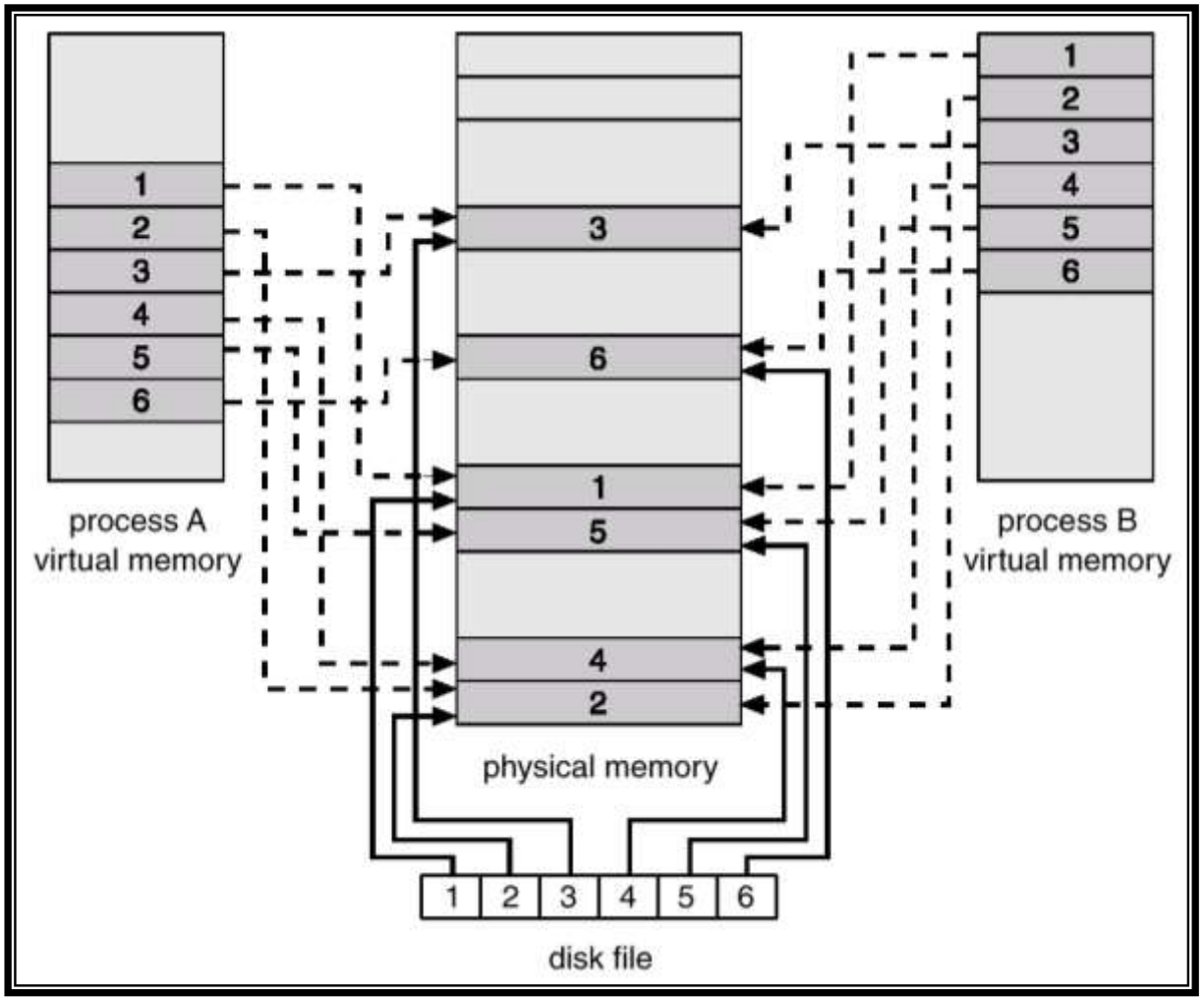
Before Process 1 Modifies Page C



After Process 1 Modifies Page C



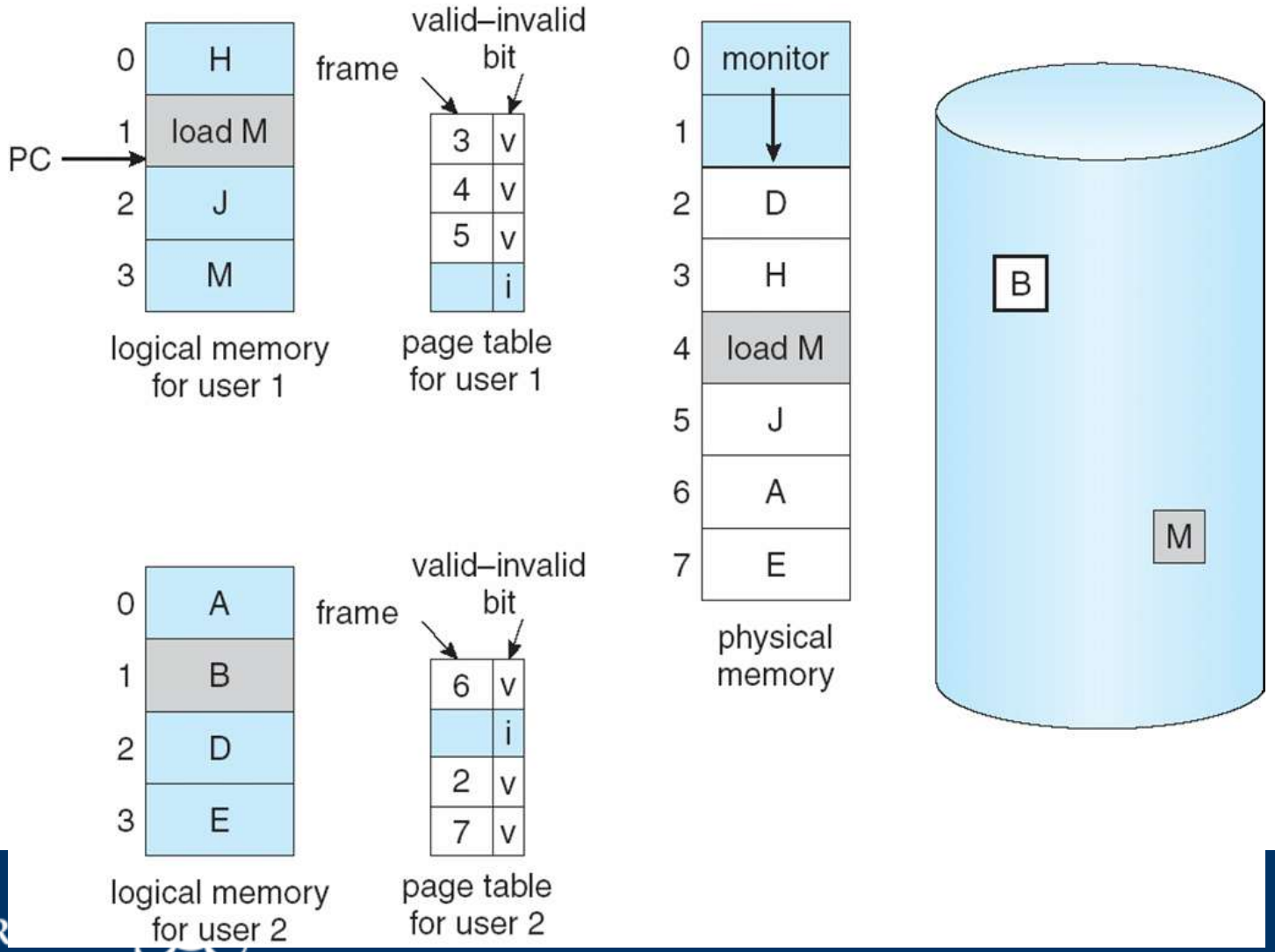
Cara Proses Creation: Memory mapped file



What happens if there is no free frame?

- Yes: Page Replacement.
- Pendekatan :
  - Jika tidak ada frame yang kosong, cari frame yang tidak sedang digunakan, lalu kosongkan dengan cara menuliskan isinya ke dalam swap space, dan mengubah semua tabel sebagai indikasi bahwa page tersebut tidak akan berada di memori lagi.
  - Bagaimana algoritmanya?

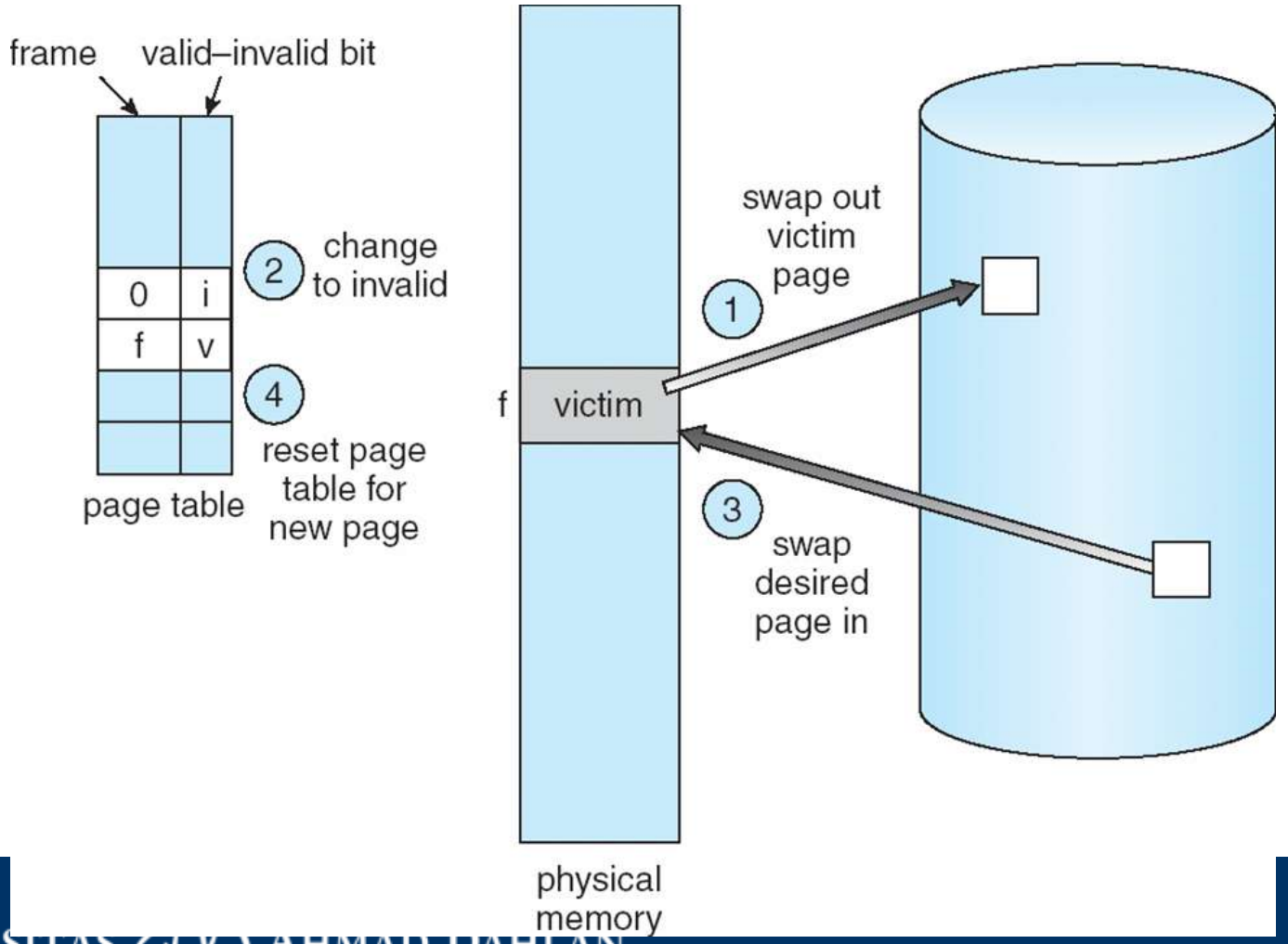
Need For Page Replacement



Yang dilakukan saat Page Replacement

- Mencari lokasi page yang diinginkan pada disk.
- Mencari frame yang kosong :
  - Jika ada, maka gunakan frame tersebut.
  - Jika tidak ada, maka kita bisa mengosongkan frame yang tidak sedang dipakai.
    - Gunakan **algoritma page-replacement** untuk menentukan frame yang akan dikosongkan.
  - Tulis page yang telah dipilih ke disk, ubah page-table dan frame-table.
  - Membaca page yang diinginkan ke dalam frame kosong yang baru.
  - Ulangi user process dari awal.

Page Replacement



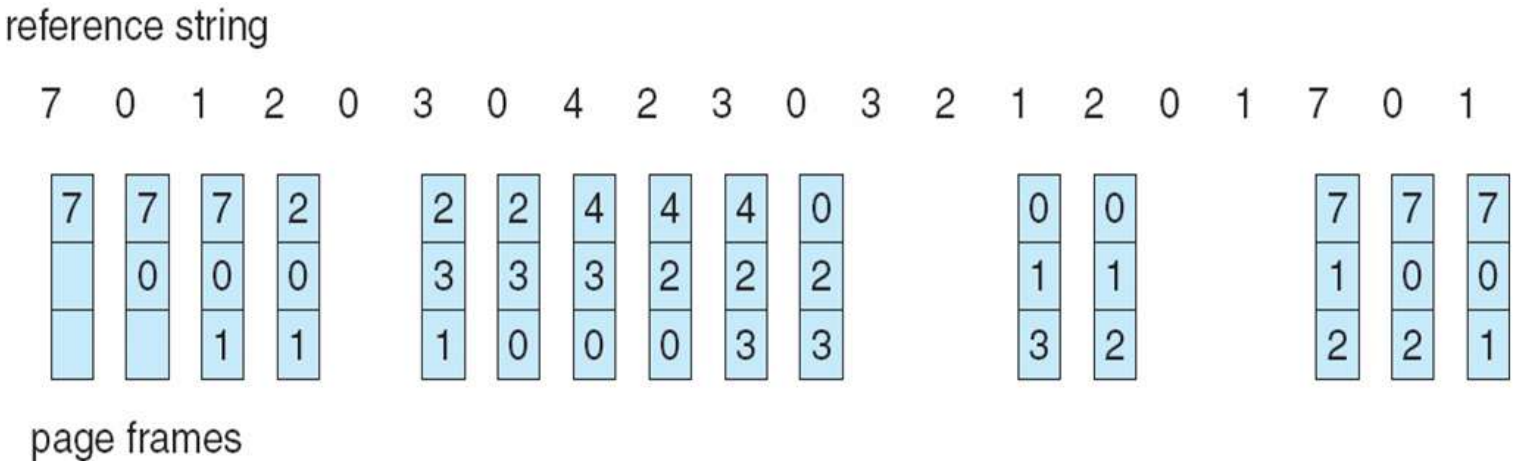
Algoritma Page Replacement

- Bertujuan untuk mendapatkan **page fault terendah.**
- Ada beberapa Algoritma Page Replacement:
  - Algoritma FIFO
  - Algoritma Optimal
  - Algoritma LRU
  - Algoritma Perkiraan LRU

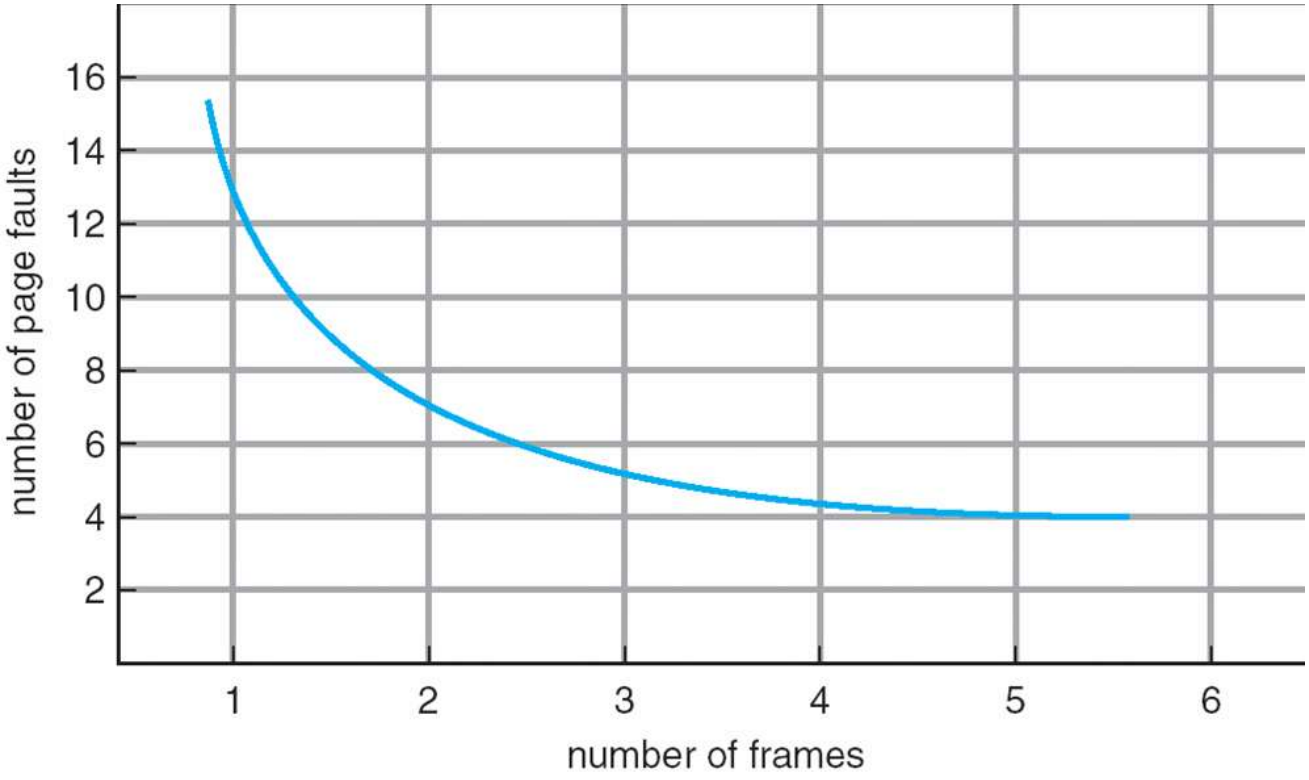
Alg. FIFO

- Page yang diganti adalah page yang **paling lama** berada di memori.
- Mudah diimplementasikan.
- Mudah dimengerti.
- Bisa mengalami Anomali Belady.
  - Page fault rate meningkat seiring dengan meningkatnya jumlah frame.
  - Hanya terjadi pada beberapa Algoritma Page Replacement.

FIFO Page Replacement



Graph of Page Faults Versus The Number of Frames



Anomaly Belady: kecepatan page fault akan bertambah jika framenya bertambah

First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)

1	1	4	5	
2	2	1	3	9 page faults
3	3	2	4	

- 4 frames

1	1	5	4	
2	2	1	5	10 page faults
3	3	2		
4	4	3		

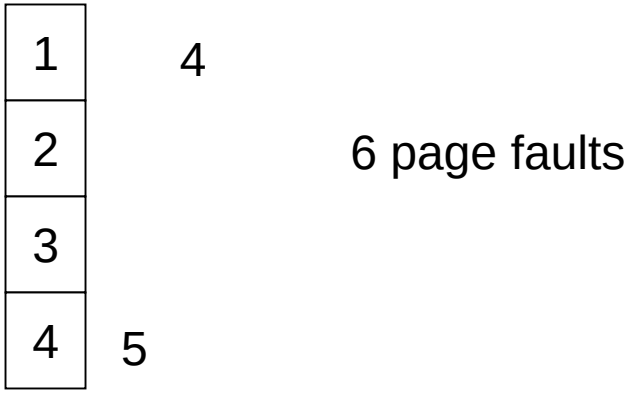
Alg. Optimal

- Page yang diganti adalah page yang **tidak akan dipakai dalam jangka waktu terlama**.
- Sulit diimplementasikan (krn prediksi sulit dilakukan)
- Memiliki page-fault terendah.
- Tidak akan mengalami Anomali Belady:
  - Tidak : more frames $\Rightarrow$ more page faults

Optimal Algorithm

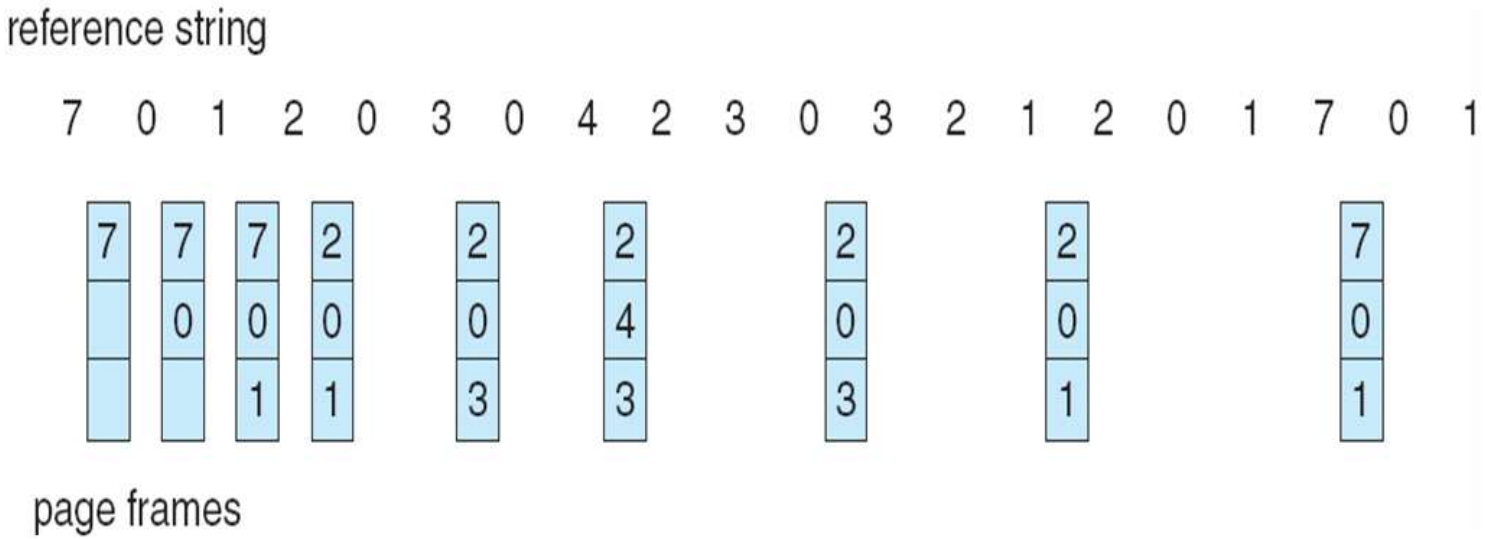
- 4 frames example

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



- How do you know this?
- Used for measuring how well your algorithm performs

Optimal Page Replacement

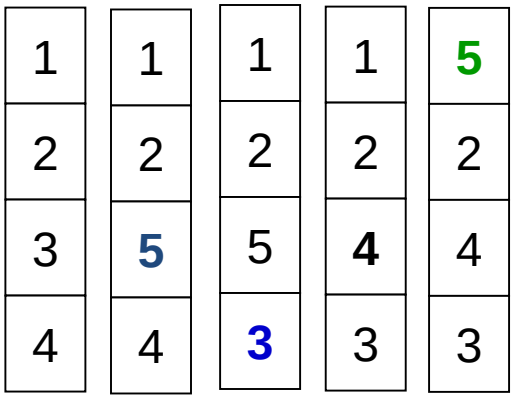


Alg. Least Recently Used

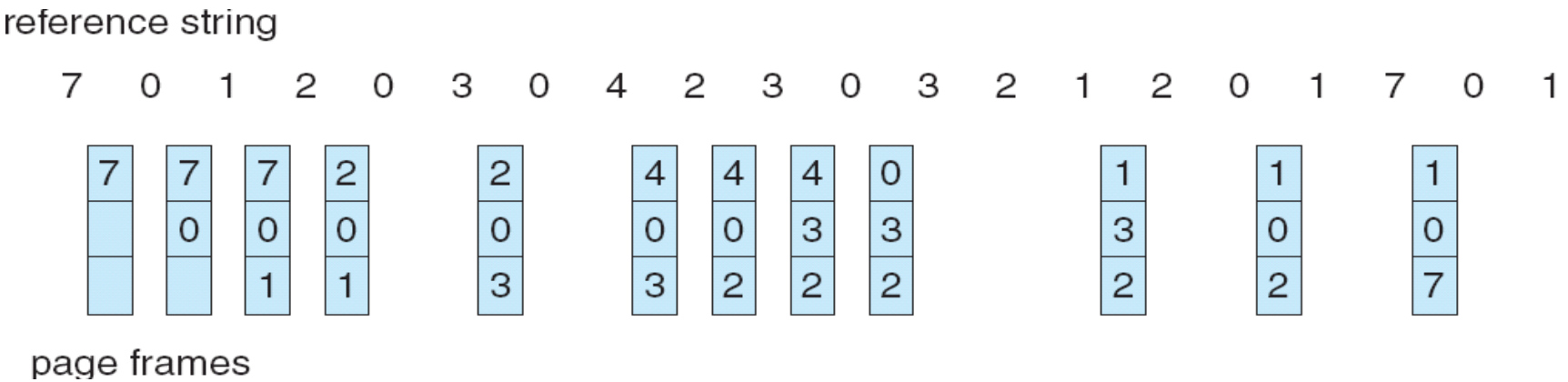
- Page yang diganti adalah page yang **tidak baru saja digunakan**.(paling lama tidak digunakan)
- Merupakan perpaduan antara Algoritma FIFO dan Algoritma Optimal.
- Sulit diimplementasikan.
- Tidak akan mengalami Anomali Belady.

Least Recently Used (LRU) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, **5**, 1, 2, **3**, **4**, **5**



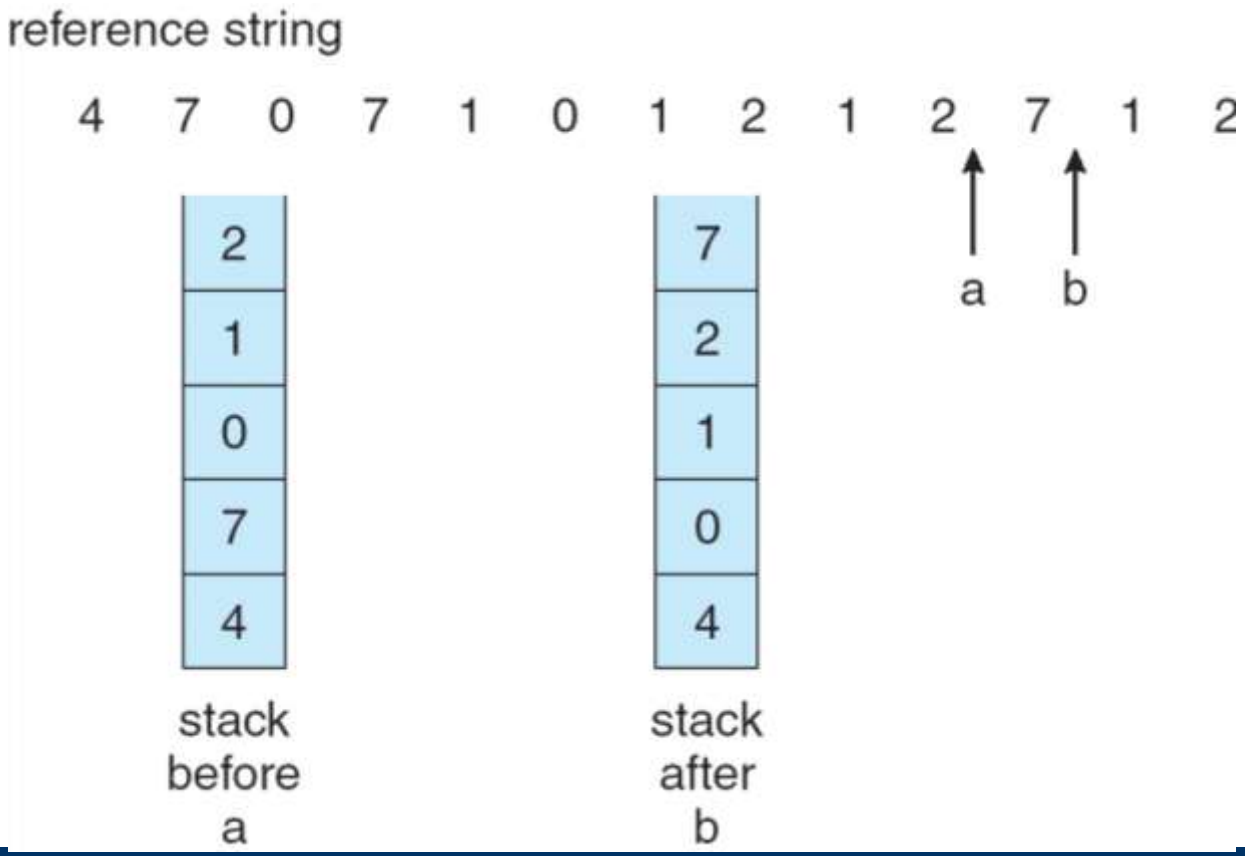
LRU Page Replacement



Alg. LRU

- Dapat diimplementasikan dengan 2 cara, yaitu :
  - Counter
    - Menggunakan clock yang nilainya akan ditambah 1 tiap kali melakukan reference ke suatu page.
    - Harus melakukan pencarian.
- Stack
    - Tiap mereference ke suatu page, page tersebut dipindah dan diletakkan pada bagian paling atas stack.
    - Page yang diganti adalah page yang berada di stack paling bawah.
    - Tidak perlu melakukan pencarian.
    - Lebih mahal.

Use Of A Stack to Record The Most Recent Page References



Alokasi Frame

- Alokasi frame berhubungan dengan mekanisme alokasi sejumlah memori bebas untuk proses-proses.
- Fixed Allocation
  - Proses dengan prioritas tinggi ataupun rendah diperlakukan sama.
    - Equal Allocation: semua sama rata
    - Proportional Allocation: sesuai kebutuhan
- Alokasi prioritas
  - Perbandingan frame-nya tidak tergantung pada ukuran relatif dari proses tetapi tergantung pada prioritas proses.

Jenis Page Replacement

- **Global replacement** mengijinkan suatu proses untuk menyeleksi suatu frame yang akan dipindah dari sejumlah frame, meskipun frame tersebut sedang dialokasikan ke proses yang lain.
- Pada **local replacement**, jumlah frame yang dialokasikan untuk proses tidak berubah.
  - Setiap proses dapat memilih dari frame-frame yang dialokasikan untuknya.

Thrashing

- Kegiatan paging yg sangat tinggi
- **Thrashing** ≡ a process is busy swapping pages in and out
- If a process does not have “enough” pages, the page-fault rate is **very high**.
- This leads to:
  - low CPU utilization
  - operating system thinks that it needs to increase the degree of multiprogramming
  - another process added to the system
- Proses menghabiskan waktu lebih banyak untuk paging daripada eksekusi.

UAS

- Open Book, tidak boleh saling pinjam
- Materi :
  - Sinkronisasi proses (Chapter 7)
  - Critical Section, critical region, atomic process (Chapter 7)
  - Deadlock,starvation (Chapter 8)
  - Memori management (Chapter 9)
  - Virtual Memori (Chapter 10)

NEXT

- File-System